Lab 06

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

CL 1004 – Object Oriented Programming

BCY

Outline

Assa

* Inheritance
* Types of Inheritance : Single, Multilevel
* Examples
* Exercise

## Object Relationship

* + Object oriented programming generally support 4 types of relationships that are:

Inheritance

* Is-a relationship

Composition

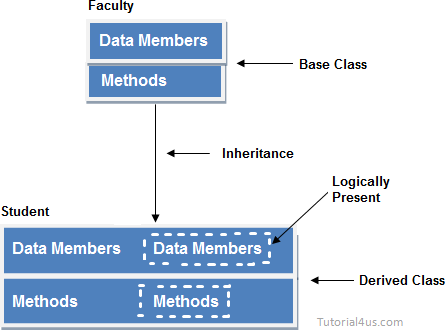
* Part-of relationship

Aggregation and Association

* has-a relationship

## Inheritance

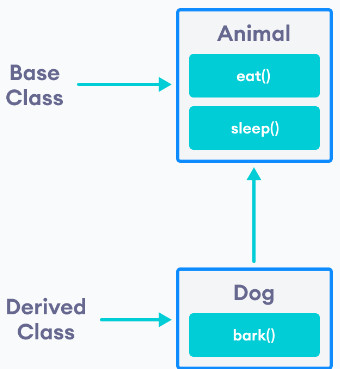
* + The technique of deriving a new class from an old one is called inheritance



* + Capability of a class to derive properties and characteristics from another class.
  + The extended (or child) class contains all the features of its base (or parent) class, and may additionally have some unique features of its own.

**Sub Class:** The class that inherits properties from another class is called Sub class or Derived Class. **Super Class:** The class whose properties are inherited by sub class is called Base Class or Super class

### For example



***Program as Example***

#include <bits/stdc++.h> using namespace std;

//Base class

class Parent

{ public: int id\_p;

};

// Sub class inheriting from Base Class(Parent)

class Child : public Parent

{

public:

//main function int main()

{

Child obj1;

// An object of class child has all data members

// and member functions of class parent obj1.id\_c = 7;

obj1.id\_p = 91;

cout << "Child id is " << obj1.id\_c << endl; cout << "Parent id is " << obj1.id\_p << endl;

return 0; }

***Program as Example***

|  |  |  |
| --- | --- | --- |
| #include <iostream> using namespace std; class Animal { public:  void eat()  { cout << "I can eat!" << endl; } void sleep()  { cout << "I can sleep!" << endl; } }; | // derived class  class Dog : public Animal { public:  void bark()  { cout << "I can bark! Woof woof!!" << endl; }  }; | int main() {  // Create object of the Dog class Dog dog1;  // Calling members of the base  class dog1.eat(); dog1.sleep();  // Calling member of the derived class |

# Output

I can eat!

I can sleep!

I can bark! Woof woof!!

### Implementing inheritance in C++

* + Syntax:

class subclass\_name : access\_mode base\_class\_name

{//body of subclass};

### Mode of Inheritance

* + **Public mode**: If we derive a sub class from a public base class. Then the public member of the base class will become public in the derived class and protected members of the base class will become protected in derived class.
  + **Protected mode**: If we derive a sub class from a Protected base class. Then both public member and protected members of the base class will become protected in derived class.

**Private mode**: If we derive a sub class from a Private base class. Then both public member and protected members of the base class will become Private in derived class

class A

{

public:

int x; protected: int y;

private: int z;

};

class B : public A

{

// x is public

// y is protected

// z is not accessible from B

};

class C : protected A

{

// x is protected

// y is protected

// z is not accessible from C

};

class D : private A // 'private' is default for classes

{

// x is private

// y is private

// z is not accessible from D

};

**Single Inheritance**

class Person

{

char name[100],gender[10]; int age;

public:

void getdata(){

cout<<"Name: "; cin>>name; cout<<"Age: "; cin>>age; cout<<"Gender: "; cin>>gender;}

void display(){

cout<<"Name: "<<name<<endl; cout<<"Age: "<<age<<endl; cout<<"Gender:

"<<gender<<endl;

}};

class Employee: public Person

{

char company[100]; float salary;

public:

void getdata()

{

Person::getdata(); cout<<"Name of Company: "; cin>>company;

cout<<" Salary: Rs."; cin>>salary;

}

void display()

{

Person::display();

cout<<"Nameof Company:"<<company<<endl; cout<<"Salary: Rs."<<salary<<endl;

}};

int main(){

Employee emp; cout<<"Enter data"<<endl; emp.getdata(); cout<<endl<<"Displaying

data"<<endl;

emp.display();

R eturn 0;}

## Multilevel Inheritance

* + In this type of inheritance, a derived class is created from another derived class.

#include <iostream> using namespace std; class base {

public:

void display1()

{ cout << "\nBase class content."; } }; class derived : public base {

public:

void display2()

{ cout << "1st derived class content."; }

};



class derived2 : public derived

{ void display3()

{ cout << "\n2nd Derived class content."; }

};

int main()

{ derived2 D;

//D.display3(); D.display2();

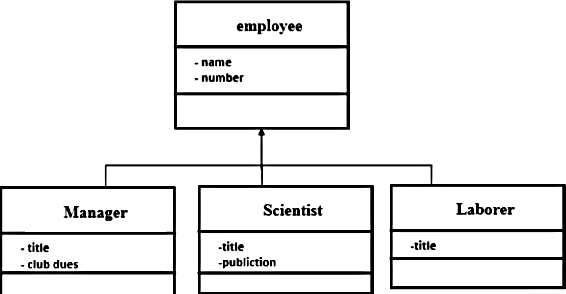
D.display1(); }

Activity

Single Inheritance:

1. Write a C++ program to define a base class Item (item-no, name, price). Derive a class Discounted- Item (discount-percent). A customer purchases 'n' items. Display the item-wise bill and total amount using appropriate format.
2. In the database only three kinds of employees are represented. Managers manage, scientists perform research to develop better widgets, and laborers operate the dangerous widget-stamping presses.

The database stores a name and an employee identification number for all employees, no matter what their category is. However, for managers, it also stores their titles and golf club dues. For scientists, it stores the number of scholarly articles they have published and their title. Laborers store the title only. You must start with a base class employee. This class handles the employee’s last name and employee number. From this class three other classes are derived: manager, scientist, and laborer. All three classes contain additional information about these categories of employee, and member functions to handle this information as shown in figure.



Multilevel Inheritance:

1. Create a class named Employee with private data members int empNo, char[20]name, char[20]des. Also give the implementation of two member functions:

* void getEmployeeDetails() that takes input from user
* employee\_display(): that displays the employee record.

Create another class Salary privately inherited from Employee having float data members basic\_pay, human\_resource\_Alowance, dearness\_allowance, profitablity\_fund and net\_pay. Also give implementation of three member functions:

* getPayDetails(): that takes all all the inputs(employee class as well as salary class)
* Calculate\_net\_pay: Netpay= bp+hra+da-pf
* Salary\_display(): Displays all data(salary class as well as employee class)

Create another class BankCredit privately inherited from Salary having int accountNo, char[20]bankname, char[20]IFSC. Also give implementation of two member functions:

* getBankDetails(): that takes all all the inputs(bank class, employee class as well as salary class)
* display(): Displays all data(bank class,salary class as well as employee class)

Instantiate the program for n number of persons( n specified by user).